CRYPT-01

crypt() is cryptographically weak; use stronger alternatives

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5354 bytes

Attack Category	• Encryption Assault
Vulnerability Category	• Cryptography
Software Context	• Cryptography
Location	
Description	crypt() is cryptographically weak and stronger alternatives should be preferred for the hashing of passwords. The standard crypt function can be used to hash passwords using a DES-based algorithm. It uses a password (up to 8 characters) and a salt (2 bytes) to generate a key to encrypt a string of zeroes using a slight variation of the DES algorithm. The problem is that the password can only be 8 characters long and there are 95 choices for each character. If a constant salt is used (which is the case if a salt is not explicitly provided to the crypt function), then an attacker can build a "dictionary" of all 2^53 possible passwords and their hashes. Determining passwords then becomes a trivial table lookup. Even if the salt is different for each password, the dictionary size will increase to 2^66. In a few years, it should be practical to create a dictionary of that size. Also note that the effective dictionary size may be much smaller, since humans tend to use passwords that are easy to remember. Only considering passwords composed of English words would be a good way to obtain passwords from a database. Hence, the crypt function should generally not be used to hash passwords.
APIs	**Function Name** / **Comments** crypt
Method of Attack	
Exception Criteria	

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

Solutions

Solution Applicability	Solution Description	Solution Efficacy
When secure hashing (e.g., of passwords) is required	Replace all calls to crypt() with a more secure version. The following libraries provide good choices: * Cryptlib (Peter Gutmann) - not free, very robust * OpenSSL - freely available, popular choice * Crypto++ - open source * BSAFE - RSA Security's widely deployed commercial library Some of the most widely available hash algorithms, though better than crypt(), may not be entirely secure. The MD5 algorithm has been discovered to have significant weaknesses, and the first hints of vulnerability have been found in SHA-1. Stronger algorithms in the SHA family are likely the most secure options at	Effective, particularly if SHA-2 variant is used.

	present, but are not supported by all crypto libraries.
Signature Details	char *crypt(const char *key, const char *salt)

Examples of Incorrect Code	`[...]`
	`hash = crypt(key, salt);`
	`[...]`

Examples of Corrected Code	`/* SHA-1 algorithm from OpenSSL library. This is the best hash algorithm currently supported by OpenSSL, but SHA-2 should be preferred for high security applications. */`
	`[...]`
	`unsigned char[SHA_DIGEST_LENGTH] hash;`
	`SHA1(passwordPlusSaltText, strlen(passwordPlusSaltText), hash)`
	`[...]`
	`/* Be aware that hash is a byte array rather than a null-terminated string as is returned by crypt().`
	`If a printable string is needed, further encoding would be required, e.g., with base64 encoding. */`

Source Reference	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch. 11.
Recommended Resources	• Wikipedia entry on MD5 algorithm[2] • Wikipedia entry on SHA-1 algorithm[3]
Discriminant Set	**Operating Systems**: • UNIX (All) • Windows **Languages**: • C • C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com